Computer Science

STUDYING THE IMPACT OF USING DYNAMIC UPDATES ON A WELL-KNOWN CLUSTERING PROBLEM

Mynul H Khan
Department of Computer Science, St. Cloud State University,
720, 4th ave S., St. Cloud State, MN 56303. Email: src16apr@hotmail.com

Zubair Shams
Department of Computer Science, St. Cloud State University,
720, 4th ave S., St. Cloud State, MN 56303. Email: znoman@eeyore.stcloudstate.edu

*Sarnath Ramnath
Department of Computer Science, St. Cloud State University,
720, 4th ave S., St. Cloud State, MN 56303. Email: sarnath@eeyore.stcloudstate.edu

Clustering of data entities is a frequently used process with several applications. A set of entities $x_1 \ldots x_n$ and the dissimilarities between all pairs of entities (x_i, x_j), where $1 \leq i, j \leq n$, can be represented by a graph with vertices $v_1 \ldots v_n$ where the length of edge between vertices (v_i, v_j) represents the dissimilarity between x_i and x_j. A cluster is a subset of the vertices, and the diameter of the cluster is the length of the longest edge e_{ij} such that both v_i and v_j belong to that cluster. Minimum sum of diameters bi-clustering problem seeks to partition the vertices into two clusters such that the sum of the diameters of the two clusters is minimized. The problem of finding a partition into three or more clusters that minimizes the sum of diameters is NP-hard. Minimum sum of diameters bi-clustering can, however, be solved efficiently by reduction to determination of the satisfiability of a 2-Conjunctive Normal Form or 2-SAT expression. Hansen provided an algorithm that solved O (nlogn) 2-SAT instances and runs with time complexity O (n^3 logn). Sarnath has provided an improved clustering algorithm that incrementally solves O (m) 2-SAT instances on a graph of m edges and runs in O (mn) time. The incremental algorithms, however, have considerable overhead due to data structures that have to be maintained. This research examines the practical trade off between this overhead and the time saved by incremental version. Both algorithms were implemented to partition instances of a complete graph having n, $50 \leq n \leq 100$, vertices. In tests of the two algorithms on these instances, the incremental algorithm was consistently observed to perform better.